OPEN TEXT CORPORATION

Product Line Technical Overview

Open Text Corporation
180 Columbia St. W., Suite 2110
Waterloo, Ontario, Canada
N2L 3L3
Tel: (519) 888-7111
Fax: (519) 888-0677

Table of Contents

1.0 Overview and Summary

Open Text's product line consists of software modules that combine in various ways to provide text database functions to end users and to other computer applications. This document provides a technical description of each of these modules, with emphasis on the technical features and the interfaces available.

Text databases have the following characteristics:

- A high proportion of the database consists of textual as opposed to numeric material.
- The data is organized into structures that are irregular in size and hierarchical in their relations.
- The data requires sophisticated presentation techniques, for example typography and page composition, to serve the needs of end users.

Examples of text databases are:

- The policies and procedures manual of a large insurance company.
- A body of text generated by applying Optical Character Recognition technology to a large number of optically captured page images.
- The inventory of articles maintained by a reference publisher for use in the creation of a variety of general and special purpose encyclopedia products.
- The body of technical documentation associated with the product line of a heavy-equipment manufacturer.
- The legislation, case law, and transcripts used in support of a large-scale litigation.

Open Text's product line contains:

- The PAT text database server.
- LECTOR, a text display client used for on-screen presentation of database contents.
- Database loading and maintenance utilities.
- A programmer's tool kit to aid in using the Application Programming Interface to the PAT Server.
- A variety of client query programs which provide end users with access to the PAT Server.
- The Parallel Execution Monitor, PEM, for parallel distributed multi-database searching

All of these products use an open, client-server architecture. For this reason, they can be combined with each other and with other program modules to construct solutions to many different business

problems.

All of the products have been exhaustively tested on very large databases and provide performance which dramatically exceeds that available from any other such system.

2.0 Open Text System Architecture

Because Open Text software was designed and built in 1989-1990, the designers of the Open Text text database software had the benefit of being aware of current market forces and using this knowledge to develop a truly next-generation text database product. In the sections to follow, we discuss the architecture and powerful search features of this system, its modular design, functionality and adherence to applicable industry standards. The advantages of the Open Text system can be divided into five general categories.

- User data is stored in an open format. The data indexed and searched by Open Text products is kept in its format prior to indexing. Users do not have be concerned about any transformations on, or encryptions of, their data.
- The Open Text search engine provides unparalleled handling of structure in a document. An important consequence is full native SGML support.
- Extremely high performance on very large databases.
- The ability to perform phrase searches without any performance loss. Searches for long phrases complete in the same time as searches for single words.
- Consistently high performance, regardless of the size of the query results set, frequency of occurrence of the text being searched for, or size of the database.

2.1 Open Text's System Architecture

The Open Text text database architecture provides for a true client-server based system. The system has been designed with a distinct division of functionality between the client and server modules, with support for:

- client and server operation on the same machine, with terminal access by users
- client and server operation on separate machines, with network communication between client and server. Network communications may include LANs, TCP/IP, WAN protocols or modem dial-up.

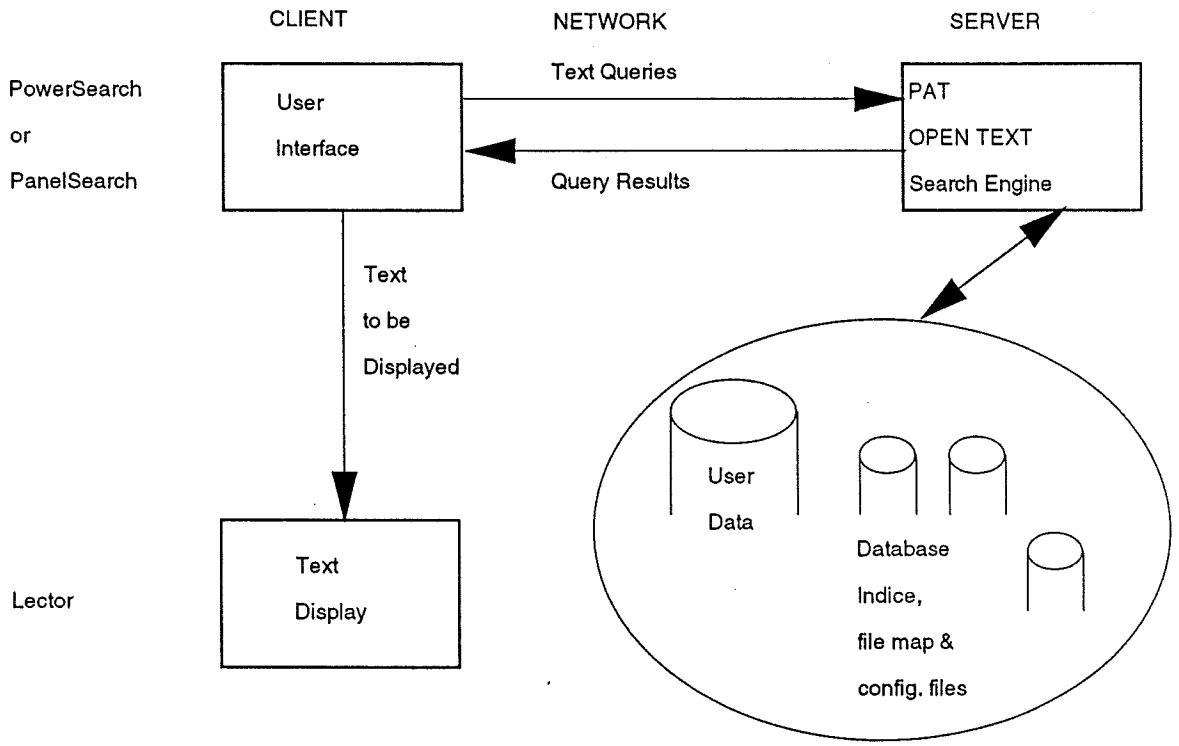
In configurations in which the client and server operate on the same host, standard communication mechanisms are used. The communication mechanism of choice on the UNIX platform is the pipe facility, while DDE and OLE are used on the Microsoft Windows platform. In addition, the message flow between client and server is rigorously and clearly documented.

The decision of whether to run Open Text client and server on the same machine or on separate machines is a system administration and configuration decision. Installations which run the client

and server components on different machines use standard network communication protocols. These include protocols include TCP/IP, UDP, Berkeley sockets, OSI TP0, XDR, NDR and UNIX rsh.

An outline of the system architecture is provided in the following diagram. The search engine functions as a server which is accessed by the User Interface module. Communication between the User Interface and the Search Engine is through the standard communication protocols mentioned above. In response to queries from the User Interface, the search engine must access the text data and indices of the text database. This access is performed through standard file system access calls, thus allowing standard third party backup and administration tools to be used on the text database.

Open Text System Overview



The display of text is handled by the Open Text client product, Lector, which receives text from the User Interface. Lector does not access the text database directly and usually runs on the client machine. As a result, Lector has absolutely no negative impact on the server's performance.

The result of this approach is a system which exposes and documents what are typically considered to be internal system communication paths. This exposed design results in tremendous benefits to the system integrator.

Integration with the Open Text system is particularly straightforward for two reasons:

- the messaging and communication paths in the system are exposed and documented
- the communication mechanisms used for these messages are standard industry standard pipes on Unix platforms and DDE, DLL and OLE on Microsoft Windows.

The approach to various integrations and the impact of this modular architecture is discussed in a subsequent section of this document entitled *Integration and the API*.

3.0 Client-based User Interfaces

The PAT Server is well-suited to supporting ad-hoc end-user access to text databases, a common objective for text database providers. Open Text provides a set of query clients for the PAT Server; all are based on the same standard API.

3.1 PowerSearch User Interface - OSF/Motif

This is a program which provides the OSF/Motif look and feel, based on the X Window System. The PowerSearch UI provides general-purpose access to virtually all of the underlying capabilities of the PAT Server; an extremely rich query environment. Some of its features are:

- Simultaneous graphical display of session history, result samples, and structures available in the database.
- Menu-driven access to all operations. The most common operations are accomplished simply by selecting objects on the screen without menu intervention.
- On-line site-customizable help system.
- Graphical feedback on query results.
- Configurable menu-driven export - when the user selects a result for display or processing, it can be transmitted to any other task on the system, as specified in a configuration file.

This program communicates with the server using Unix pipes and sockets.

Despite its power, PowerSearch is intuitive and easy to use. The experience of Open Text's customers is that non-technical end-users can quickly learn to accomplish sophisticated compound queries using PowerSearch.

3.2 PowerSearch User Interface - Microsoft Windows

This program duplicates the functionality of OSF/Motif PowerSearch, but operates in the MS Windows (3.1 or higher) environment.

This program communicates with the PAT Server through a DLL interface. Open Text provides DLL modules which support WinSock compatible network interfaces. These include modem support with PPP. These modules are available in source code form to those who wish to integrate with other network architectures.

The MS Windows PowerSearch facility can be integrated to other tasks, for example for display, printing, or editing, through a DDE interface.

3.3 SGML Query for OSF/Motif and Microsoft Windows

In the case where the database contains validated SGML, it is sometimes desired to phrase database queries in terms of SGML objects such as elements and attributes. While the facilities of the PowerSearch user interface modules provide all the facilities necessary to accomplish these queries, they also go to some length to hide the intricacies of the underlying SGML mechanisms.

In this case, if the indices have been constructed with the use of the SGML Preprocessor discussed above, the SGML Query module is available for both the OSF/Motif and Microsoft Windows user interfaces. This module provides an additional pop-up window that automates the construction of queries with direct access to the SGML elements and attributes.

3.4 PanelSearch for Microsoft Windows

This is a graphical interface which operates in the MS Windows (3.1 or higher) environment.

PanelSearch provides access to only a subset of the capabilities of the PAT Server, but in a greatly simplified fashion. Unlike the other modules, PanelSearch for Windows assumes the existence of a fundamental record type, and hence supports traditional simplified Boolean AND, OR, and NOT at the record level. (The PowerSearch modules support a much more flexible and powerful set of logical operations).

3.5 PanelSearch for character mode Users (ASCII terminal emulator)

This program executes in the Unix environment and duplicates the functionality of the Microsoft Windows PanelSearch. However, its user interface, while menu-driven, is character-based, rather than using GUI Window/Icon/Mouse/Pointer technology. This means that access to PanelSearch for character mode requires only an ASCII terminal or, more commonly, a terminal emulation program such as for VT-200. Such emulation programs are commonly available for all known personal computer and network configurations.

PanelSearch for character mode is very valuable in situations where database access is required from a large number of incompatible types of workstations and personal computers, particularly in the case where dial-up access from remote computers is desired. While it may be difficult or impossible to enforce a common GUI or network architecture, it is usually practical to arrange for terminal-emulator access to the database server.

While not as visually appealing as the OSF/Motif or Microsoft Windows query interfaces, the PanelSearch for character mode offers high performance and is easy to use.

3.6 The LECTOR Text Display Client (OSF/Motif and Microsoft Windows)

LECTOR is a program which displays database text on the screen in real time and with attractive user-specified formatting. LECTOR's formatting operations are based on a *Display Specification*, which can be thought of as a style sheet.

The Display Spec associates *tags* and *attributes* with display characteristics such as color, font size, font family, font weight, highlighting, text suppression, tag suppression, indenting and tabbing. A tag is defined as a string of characters with a unique start character that is not otherwise used in the text. An attribute is defined in the SGML style.

A LECTOR Display Spec typically contains several *formats*, which can be thought of as different style sheets for the same data. LECTOR allows the user dynamically to switch between formats while the text is on the screen.

LECTOR can display text received from another program, for example PAT, or can display text read directly out of a file. LECTOR has facilities for exporting text selected by the user (with the mouse) to another program. This has been used to implement a variety of database navigation schemes and hypertexts.

LECTOR supports dynamic resizing of the display window and changes in the number and width of columns. It also provides selection and copy operations, so that text can be cut out of the LECTOR window and pasted into a word processor, search, or other application window.

There is also support for direct access to special characters in the display font based on recognition of different tag/attribute combinations.

Perhaps most important, LECTOR is very fast, rendering text onto the screen with near-instant response. For this reason, a database is often implemented with LECTOR and another viewer. In this scenario, LECTOR is used for initial database browse, and the other viewer, presumably one with different display facilities but which may run much slower, is used for exact WYSIWYG presentation of data that has been determined to be of interest after initial browsing with LECTOR.

4.0 The PAT Text Database Server

The PAT text search engine is central to the Open Text system. It is implemented as a process in the Unix environment and as a Dynamic Link Library in the Microsoft Windows environment. The interface to the search engine is a platform independent query language oriented API. This API is discussed in a subsequent section of this document, entitled *Integration and the API.* In this section we will discuss some of the capabilities of the PAT search engine and provide a more detailed view of the engine architecture.

4.1 The Text Database

Similar in concept to traditional databases containing record oriented numerical data, text databases consist of bodies of related texts which are available for searching. The criteria for grouping different texts together in a database is to a large extent dependent on the application criteria and may be based on the content of the data (e.g. engineering, product support, accounting, etc.), the data format (e.g. SGML, WordPerfect, RTF, FrameMaker, etc.) or other requirements.

We will now provide an overview of the text database and its primary components. The I/O filter subsystem which allows native searching of the various data formats is described in the next subsection of this document.

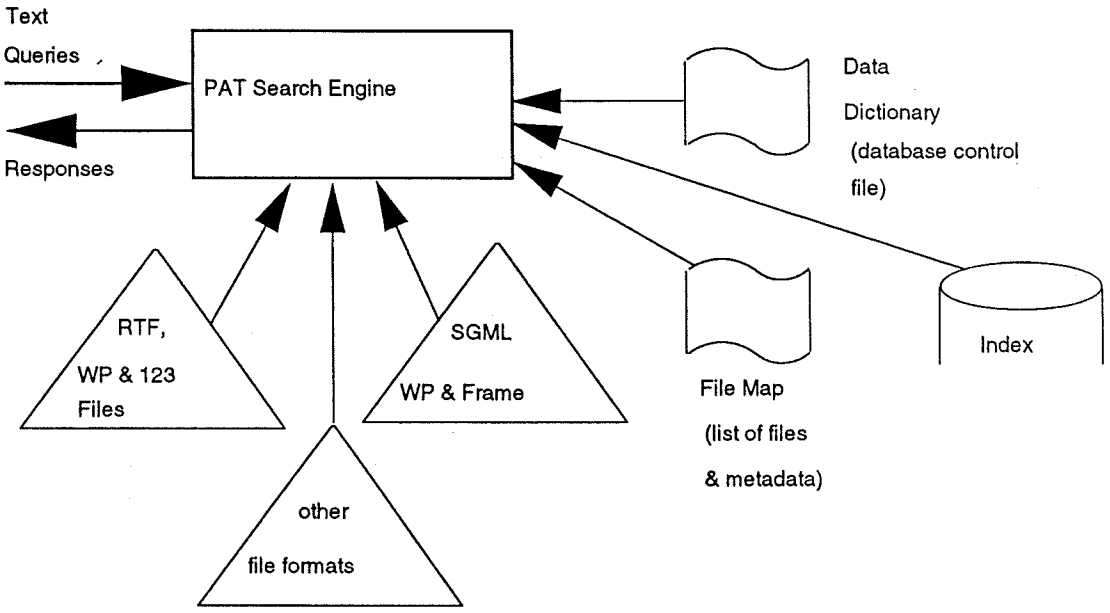
A text database consists of

- the text to be searched
- metadata associated with this text
- control files which provide system integrity and administration

These components and their relationships are illustrated in the following figure. The PAT search engine receives queries from the client user interface and returns responses. All the remaining modules in the diagram are files which contain portions of the text database. The triangular pieces represent portions of the file system directory hierarchy which contain files to be searched by PAT.

The primary control file which lists all the database components is the Data Dictionary. This ASCII file, which contains references to the other components of the database, is the heart of the system. The text which is being searched by PAT can be spread across any number of files. These files may be in a multitude of data formats.

PAT Search Engine a Multiple File Format Database



Directory Hierarchy Containing Files
of Various Formats

The file map is an ASCII text file which lists:

- all the files which contain the text in the database
- the format of each file
- an arbitrary and unlimited amount of descriptive information about each file (metadata)

In order to maintain a modular approach and access the data in its original format, the metadata and filemap information is kept physically separate from the main text data. However, for search purposes, the filemap information (including the metadata) is logically included as part of the text database. The metadata pertaining to each file is logically located at the beginning of that file.

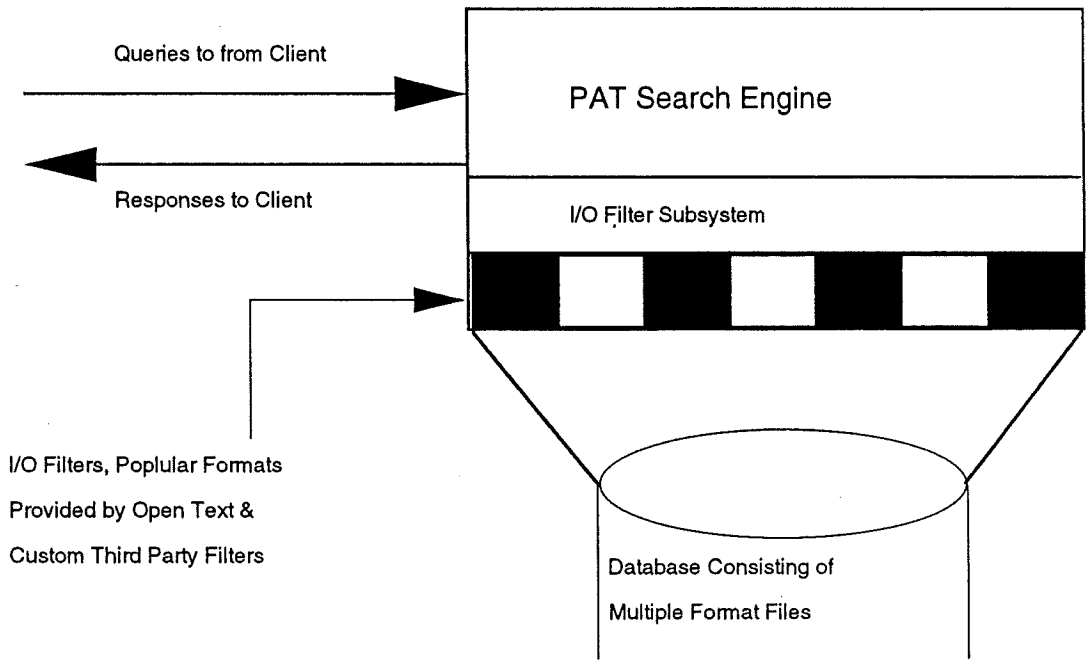
4.2 Native Data Format Support and the I/O Filter Subsystem

The PAT search engine provides native support for multiple data formats through the I/O filter subsystem. The I/O subsystem is a modular framework for filters which understand various file formats. Open Text provides filters for most popular word processor and spread sheet formats.

Integrators and developers who wish to write their own filters to search other data formats not

PAT Search Engine with Multiformat, Multifile

I/O Filter Subsystem



supported by Open Text may do so. The interface to the I/O subsystem is defined and published by Open Text, along with source code for sample filters.

These filters provide basic C language "stdio" functionality to PAT and incorporate the ability to react properly (for example, ignore) special control codes that are unique to the format being searched.

4.3 Index Files

The PAT Server attains its high speed by searching through Index Files rather than scanning the whole text. Most important is the string index file, used to support the basic string searching operations. This can only be generated by Open Text's PAT Index Builder. This file is typically 50 to 75 percent of the size of the original data file.

It is also common to pre-build indices for commonly-used structures in the text. These indices consume about 8 bytes of space for each instance of each structure. The format of these files is simple and fully documented. While Open Text provides tools to build these files, customers have on occasion written their own programs to create specialized structure indices.

4.4 Server Features

The following is a list of some of the features included in the Open Text PAT search engine:

- native SGML support
- powerful structure handling and set membership operators
- thesaurus support
- ranking support
- structure/element extraction support
- boolean operation support
- left & right proximity operators
- right only proximity operator
- statistical sampling
- query reuse and history list features
- memory management support
- Search for strings, stemmed prefixes, alphabetic ranges and patterns that frequently follow others.
- Result set naming, sampling, traversal, and display in various formats.
- Union, difference, and intersection (exact or with variable proximity) of result sets.
- Structure traversal by inclusion and containment of result sets, dynamic structure creation, ranking of structures by contained match count.
- Direct database addressing.
- "Save to" and "reload from" named files in a variety of formats.
- Macro processing and command files.

5.0 Parallel Execution Monitor (PEM)

The Parallel Execution Monitor (PEM) is a software module which allows several different PAT databases, possibly located on different computers in a network, to be created as a single database.

The PEM operates by reading a configuration file which names all the databases and the computers on which they reside. The PEM uses this information to start one copy of the PAT Server for each of the databases, automating the tricky programming task of starting programs locally and remotely and of setting up the necessary communications channels.

The user and program interfaces to the PEM are identical to those of the PAT Server; the PEM automates the task of sending the commands to all the different PAT Server processes and collating the results. Since the interface is the same, this means that any client of the PAT Server, whether from Open Text or user-written, can use the PEM for access to parallel distributed databases without any change.

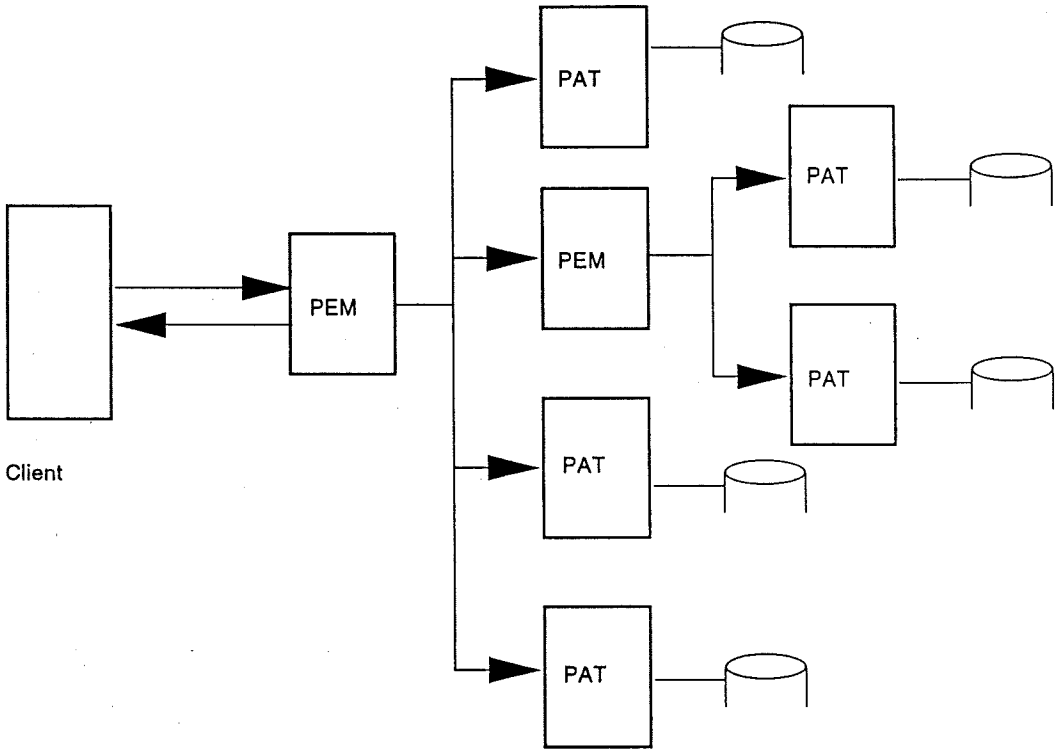
The PEM is useful in two different scenarios:

- When a database grows too large to fit on a single disk or even on all the disks attached to a computer system, the PEM provides high-performance access by distributing the data across systems.
- An organization may operate different text databases maintained independently by different groups. The PEM allows users to access all these databases in parallel as if they were one, without requiring any special co-operative effort from the database maintainers.

The PEM system is carefully designed to minimize the message traffic between the PEM and the PAT Servers. This means that it does not saturate networks. Furthermore, the application of parallel technology means that databases can grow to **very large** sizes with **no** degradation in response time, if there are sufficient processors available in the network.

The current implementation of the PEM relies on the Unix *rsh* facility, and can operate on any network where this connectivity is available.

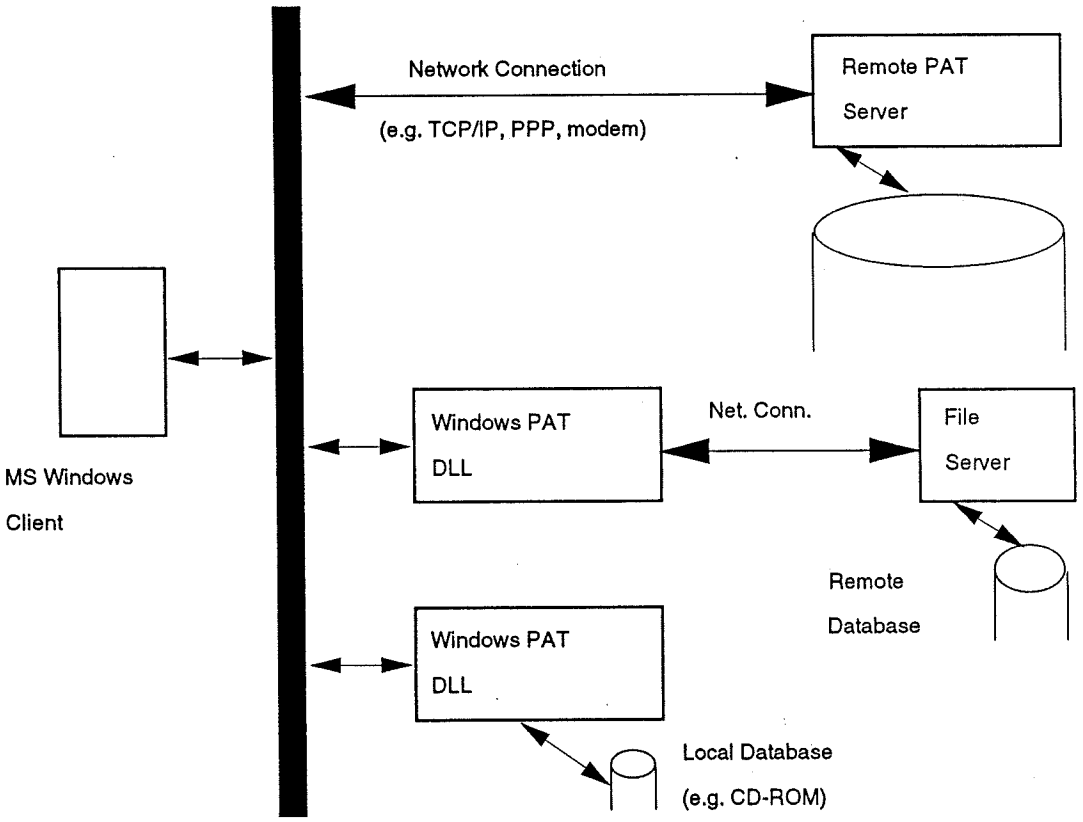
Parallel Database Search with the Parallel Execution Monitor



6.0 Integration and the API

Open Text provides a unified API for client user interfaces accessing the PAT text engine. This subroutine library API can access:

- the Unix pipe connection, for Unix clients accessing Unix based PAT servers running on the same Unix machine
- the remote shell (rsh) protocol, for Unix clients accessing Unix based PAT servers running on the a remote Unix host.
- the remote shell (rexec) protocol, for MS-Windows clients accessing PAT servers running on a Unix machine.
- the Windows PAT DLL, for MS-Windows clients accessing a local PAT engine with either a local or remote mounted (e.g. Novell) database.



6.1 Standard Communication Protocols

The communication paths between the various modules of the Open Text system follow the standards of the operating environment in which the system is installed and functioning. This allows standard programming techniques and tools of the environment to be used in implementing filters which intercept inter-module communication. For example, in the UNIX environment, inter-module communication is performed by using the UNIX pipe construct. This allows the natural use of the standard UNIX practice of connecting filters into pipelines via standard input and standard output connections by the UNIX system integrator.

In the Windows environment, the communication interfaces are based on DLLs for a high performance/high traffic interface to the search engine, and on DDEs for the GUI/Lector interface. This allows the Windows based integrator to use standard platform tools such as Visual Basic and ObjectVision to develop interfaces and filters, as necessary.

6.2 Standard Message Formats

The format of messages between the different modules of the Open Text system is fully defined and illustrated in the Open Text documentation. All communication between modules is in ASCII text, allowing for ease of debugging and the use of standard text filter technology in any integration effort. Examples of such standard filters include the typical UNIX tools lex, awk and sed. This ASCII inter-module communication is in the form of tagged SGML-like messages. This tagging is also described in the Open Text documentation and allows messages to be unambiguous, easily parsable and self descriptive.

6.3 Intercepting Communications and Replacing Modules

In order to maximize the configuration flexibility of the system and allow optimum adaptability to multivendor environments, Open Text allows for a large degree of freedom in modifying the system. The options for integrating with, or altering, the Open Text system fall into two categories:

1. replacing functional modules such as the User Interface, (Lector) text display (and even the (PAT) search engine).
2. intercepting inter-module communication. The results of such interceptions can be used to drive imaging tools, relational databases, email, faxes or virtually any other third-party application.

The "plug and play" design of Open Text allows integrators to replace the Lector text display product with the display engine of their choice, write alternative GUIs (and even choose alternative search engines).

Interception of inter-module communication allows other applications to be triggered based on

search results and queries. As an example of this integration capability, Open Text provides source code for filters between the user interface and Lector which launch imaging applications and display images. These sample filters choose the image to be displayed by triggering on information in the data retrieved. These filters may be modified and used as a basis for interacting with any other application such as email, relational database queries, etc.

Open Text also provides filters which intercept user interface-to-search engine communication and create query logs and audit trails. These can be used for billing/audit purposes. Similarly, integrators can write their own filters to monitor query/result traffic and react to it in any way desired.

6.4 The Query Language

The task of integrating a GUI or other application to the PAT Server has three parts: setting up communication with the server, generating commands for it, and interpreting the results from the server. While it is perfectly possible to accomplish all this using only the system documentation, Open Text's API Tool Kit is designed to simplify and automate each of these activities.

6.4.1 Establishing Communication

The PAT Server runs as a process, receiving commands and transmitting results. To establish communication from another program it is necessary to start up the PAT Server in such a fashion that the other program can transmit the queries and receive the results. Often it is necessary that the PAT Server be started on a remote computer in the network. The API Tool Kit contains working, debugged ANSI C source code which accomplishes this for the PAT Server on a remote or local Unix computer. When Windows/NT is commercially available, the Tool Kit will also include the appropriate source code for this environment. This source code, while proprietary to Open Text, is provided on a no-royalty basis to free it for use in other applications.

6.4.2 Generating Queries

Once a search has been requested, either by a user via a GUI or by another program, it is necessary to express that request in the syntax of the PAT Server's command language. Open Text provides integrators and developers with working source code which does this for a variety of types of query.

The query language is completely documented in a *Tutorial* and a *Reference Manual*. These between them provide a complete and comprehensive description of the facilities of the server and how best to use them via the API.

An example may be useful in providing a feel for the language.

```
Pat Text Database System, Release 4.0
Copyright 1987-1993 by
Open Text Corporation and University of Waterloo
      >> "U.S." + Americ
        1: 3944 matches
      >> 1 within region Headline
        2: 220 matches
      >> region Story including %
        3: 217 matches
      >> region DateLine within %
        4: 164 matches
      >> sample.8
        5: 8 matches
      >> pr.region.DateLine
         932460, ..<DL>WASHINGTON</DL>..
        1920436, ..<DL>WASHINGTON</DL>..
        2734200, ..<DL>UNITED NATIONS</DL>..
        3479669, ..<DL>BRUSSELS</DL>..
        4401065, ..<DL>WASHINGTON</DL>..
        5681907, ..<DL>DETROIT</DL>..
        6562676, ..<DL>TORONTO</DL>..
        7162228, ..<DL>TORONTO</DL>..
```

Many of the most commonly used features of the language are illustrated in this example, including string search and structure traversal.

6.4.3 Using Results

The previous example generated output in a human-readable format. The PAT Server can also provide results in a highly-compressed tagged-text format suitable for use by other programs, as shown below:

```
"U.S." + Americ

<SSize3944</SSize

1 within region Headline

<SSize>220</SSize>

region Story including %

<SSize>217</SSize>

region DateLine within %

<SSize>164</SSize>

sample.8

<SSize>8</SSize>

      pr.region.DateLine

<RSet><Start>932460</Start><End>932479</End>

<Raw><Size>19</Size><DL>WASHINGTON</DL></Raw><Start>1920436</
Start><End>1920455</End>

<Raw><Size>19</Size><DL>WASHINGTON</DL></Raw><Start>2734200</
Start><End>2734223</End>

<Raw><Size>23</Size><DL>UNITED NATIONS</DL></Raw><Start>3479669</
Start><End>3479686</End><

Raw><Size>17</Size><DL>BRUSSELS</DL></Raw>

<Start>4401065</Start><End>4401084</End>

<Raw><Size>19</Size><DL>WASHINGTON</DL></Raw><Start>5681907</
Start><End>5681923</End>

<Raw><Size>16</Size><DL>DETROIT</DL></Raw>

<Start>6562676</Start><End>6562692</End>
```

```
<Raw><Size>16</Size><DL>TORONTO</DL></Raw>

<Start>7162228</Start><End>7162244</End>

<Raw><Size>16</Size><DL>TORONTO</DL></Raw></RSet>
```

The Tool Kit contains object libraries and working, available source code modules which aid in parsing and interpreting this output format.

7.0 Database Maintenance and Loading Utilities

7.1 The PAT Index Builder

The PAT Index Builder is an extremely fast and robust program. For text databases which can be fit entirely into the computer's main memory, it runs at speeds of several megabytes per minute.

For larger databases, the elapsed index building time is a *quadratic* ($O(N^2)$) function of the ratio between the database size and main memory size.

This is a completely automated and hands-off operation, with built-in check point/restart facilities. For example, should a power failure occur during a lengthy index building process, the Open Text build utility will automatically restart the process at the point of failure while maintaining full index integrity.

7.2 Index Points, Special Characters, and Internationalization

As noted above, the PAT Server searches for matches to strings of characters, rather than to words. This provides important performance and functional advantages, and introduces an interesting database configuration problem. The problem is that of determining where to start the string-matching process; that is to say, where to put PAT's *index points.*

For example, in a search for the two-character pattern "id", one would expect to retrieve the word "idea"; would one also expect to retrieve "video"? This depends on whether index points are placed only at word-starts or at every character.

The PAT Index Builder is provided with a facility to allow database-specific identification of index points. This works extremely well for Western European languages, such as English, French, Italian, and Spanish, which make limited use of compound words and have small alphabets. This facility provides accurate automatic placement of index points at word and word-fragment starts and has proved very satisfactory in practical use.

However, in languages which have many compound words, such as German, the placement of index points is less obvious. The problem is even more difficult in the Asian languages, which use very large alphabets and often do not separate words typographically.

The PAT Index Builder Internationalization facility exists to solve this problem. This facility allows the user to replace the built-in index point determination features with programs custom-created for this purpose.

Open Text has established relationships with providers of Internationalization modules which perform excellent index point location for German and Japanese, and is in the process of creating

modules for other widely-used languages.

7.3 Pattern/Tag Structure Builders

In the very common case where text structures are bounded by known patterns, and the special case where these patterns are SGML-style "tags", Open Text provides programs which quickly, efficiently, and automatically create the appropriate structure indices.

7.4 Efficient Additions and Deletions of Text (patmaint utility)

New data can be added to the database without re-indexing the database. The patmaint utility can be used to add data to the database by incrementally indexing the new data only. The resulting database, after the append operation, is a complete PAT database and is not fragmented in any way.

Likewise, portions of a database can be deleted using the patmaint utility. This utility takes as its input position offsets for a series of intervals in the database and removes these segments from the database. The delete and append operations are performed in a single pass over the database. The result is a fully fledged PAT database.

7.5 SGML Preprocessor

The PAT Server provides high-performance built-in support for SGML structures. In the case where the SGML data is stored in "canonical" or fully-expanded form, the Structure Index Builder can create indices simply and efficiently.

However, when the file is real SGML - validated against a Document Type Definition (DTD) - that DTD contains valuable information about the structures to be found in the text. Furthermore, SGML allows omission or shortening of tags which mark the structures.

To aid in processing real SGML, Open Text provides a facility which reads a DTD and processes text against it, automatically writing indices to all the elements and attributes in the text.

This enables simple one-step loading of any SGML instance into a ready-to-use database.

8.0 Summary

This paper is intended to provide a technical introduction to our product architecture. It is not intended to be an exhaustive review of all the features and options of the Open Text product set.

For an increasing number of companies, Open Text technology represents the next generation of text database management.

APPENDIX - Frequently Asked Questions

This section contains responses to typically encountered requirements. The questions are presented in an order common to many Request for Proposal documents.

1.0 General Requirements

1.1 The system must be extensible and provide an API that can allow for enhancements or custom features.

The Open Text system provides an API to the PAT search engine which can be used to customize the system and develop additional features. The PAT search engine API is a query based interface. Client programs, such as user interfaces, assemble queries and send them to the PAT engine and receive the responses that are returned from PAT.

The PAT query language is well documented in a reference manual and tutorial. The connection to the PAT engine is a pipe on Unix platforms and a DLL on Windows platforms. Open Text also provides a library of routines to parse the results from the PAT engine. These routines are available at no extra cost in source form in order to allow developers to modify or study them if necessary.

The user interfaces provided by Open Text also use this same query language based API. It is important to note that all search engine functionality is accessed through this API and available to application programs such as user interfaces.

1.2 The documents must be saved in an open file format. The system must not transform the text into its own proprietary format.

When data is "loaded" for searching by the PAT search engine, indices and control files are built to support searching of the data. The text to be searched is not altered at any time during this process; it is maintained in its original format. Open Text's tools require read-only access to the text and do not make any special "internal system copies", the text is searched in its native format and database indices are stored in files which are separate from the text. This approach also makes data preparation fast and easy.

By maintaining the text data untouched, in its original form, other utilities with proper permissions may access the text without going through Open Text software. That is, Open Text's software does not need to "own the data".

1.3 The capability to search across databases must be present. This parallel multi-database search must accommodate heterogenous databases and single databases containing multiple formats.

Open Text provides a Parallel Execution Monitor (PEM) which allows the searching of multiple databases in parallel. The PEM receives queries from the application program (e.g. the GUI) and broadcasts them to PAT engines, each running on a separate database. The PEM then collates the results arriving from the various PATs and presents them in a consistent manner to the application.

The various databases searched by the PEM may be heterogenous. The data stored within a single database may also be heterogenous. So, a single database may contain various documents different formats.

It is important to note that the API for the PEM is identical to PAT and as such, its use imposes no requirement for any re-programming.

1.4 The data and application must operate in a distributed configuration.

The data must be spread over multiple disks, with the ability to keep the data and indices on separate disks.

The PAT engine accesses all text data through standard file system mechanisms. This allows the data to be distributed across multiple disks and machines, with no constraints on its layout. The engine also accesses the indices as files, thus allowing them to be distributed on different drives and machines. Furthermore, it is important to realize that during a full text search, the PAT engine restricts its activity to accessing the index. It is only when text is to be retrieved that the PAT server accesses the text files. This allows system configurations which place the indices on magnetic drives (for frequent rapid access) and text data on slower bulk storage (e.g. optical drives) for "load balancing" resulting in optimal system utilization..

The components of the system must be readily distributed across several processors.

The client (user interface) processes may be configured to run on a machine other than that which the server is running on. The server may run on one of the many different supported Unix machines, while the client runs on a Unix machine or Windows PC.

Communication between these modules running on different machines is via TCP/IP. TCP/IP implementations supporting the "winsock" standard are supported under Windows.

The system must allow parallel searches across different processors.

The Open Text Parallel Execution Monitor (PEM) allows searches to occur in parallel across multiple databases. These databases may reside on different machines. Communication between the PEM and the PAT engines assigned to each database is through the TCP/IP protocol. This

parallel search functionality is directly supported by Open Text server software and does not require API programming. As discussed earlier, the PEM receives queries from the user interface and broadcasts the queries to various PAT engines (that may be running on different processors). The results are then received from the PATs, collated and returned to the user interface.

1.5 What are the limitations on length and complexity of queries.

There are no limitations on query lengths, number of queries or complexity of queries.

1.6 The system thesaurus must be customizable.

Open Text provides a default thesaurus. This thesaurus is stored in a file in ASCII format and may be modified, customized or replaced with the thesaurus of choice. The system supports both a system-wide thesaurus as well as individual user thesauri.

1.7 How can data be added to the text database?

Data can be added to the text database through a set of utilities provided by Open Text. These utilities can be invoked interactively or in batch mode. By placing the update functionality in modular utilities, the script invocation allows updates to the database to be driven by a data stream. The source of this data stream is transparent to the PAT database; sample applications include modems, news feeds or spool areas with data emanating from an arbitrary external source.

2.0 Supported Servers/Clients

Supported Servers:

- Sun Sparc (Solaris 2.x, Solaris 80x86 and SunOS 4.1.x)
- DEC MIPS (Ultrix)
- DEC Alpha (OSF/1)
- SGI
- SCO
- RS6000 (AIX)
- HP 9000 (UX)

Supported Clients:

- OSF Motif (Same as server platforms)
- Microsoft Windows 3.1
- ASCII Terminals

This list is growing constantly, so it is best to call Open Text to check as to availability for your system. In general, Open Text is prepared to port the PAT search engine to a standard Unix environment within sixty days of acceptance of an order by Open Text head office.

3.0 Storage Requirements

The overhead for indexing data with the Open Text search engine is approximately 70% of the size of the text. This percentage varies as a function of granularity of indexing and complexity of the data structures.

3.1 What is the trade-off between compression/decompression and search time?

All Open Text indices are automatically compressed and present no decrease in search performance. Text can also be compressed to approximately 25% of its original size, thus resulting in a net 0% overhead for indexing. Compressing the text has no effect on search time since, as described earlier, the PAT engine does not access the text for purposes of searching. Only the indices are accessed, and the text is decompressed only when data is to be retrieved for viewing. Since the compressed data allows more information to be transferred per block disk read and since the decompression algorithm keeps up in performance with data transfer from the disk, the result is that the decompression overhead is offset by the faster data transfer which results in no appreciable performance degradation in retrieval times.

3.2 How can compressed data be retrieved without going through the PAT search engine

Open Text provides efficient utilities which may be invoked as sub-routines to retrieve and decompress compressed text.

3.3 Can the text data be stored off line?

Since the PAT engine requires access to only the indices in order to satisfy search requests, it is

possible to store only the indices on line, with the text information being stored off line. In this scenario, the searches will complete rapidly and the user will only have to wait for slower off line storage when requesting a document(s) for display/browsing.

3.4 Can fields be defined within a document?

Open Text allows an infinite number of fields, each of infinite length, to be defined within a document. These fields may also be nested to an unlimited extent. In fact, the PAT search engine has a consistent mechanism for treating all document structures. All structures, including the document itself and various fields such as titles, bylines, paragraphs, etc., are treated as regions. The region is an arbitrary interval of text which can be used as the basis for searching and retrieving text.

This powerful concept of regions is also the basis for PAT's native SGML support (Standard Generalized Markup Language) support.

3.5 Must all documents have values for all fields?

No. Different documents may contain different subsets of fields. Some documents may have fields with no values defined for them.

3.6 Can the list of fields and stop words vary on a per database basis?

Yes. Different databases may have different fields and different stop words defined for them.

3.7 Can fields be hierarchically nested?

Yes. Fields may be nested arbitrarily deep, so documents may contain chapters which contain sections. Sections may contain subsections, which in turn contain paragraphs. This process of nesting may continue indefinitely.

3.8 Can searches be restricted to fields?

Yes. Searches can be restricted to fields, to an arbitrary extent. For example, this allows users to request the bylines for all newspaper stories that have headlines which mention "cars" and datelines of "New York" and lead paragraphs which mention "scandals". Users may also query

based on the presence/absence of structural elements, for example: request all stories which do not contain byline fields.

3.9 Can there be multiple occurrences of the same type of field in the same document?

Yes. A document may have any number of occurrences of the same type of field.

3.10 Can field/element types within a document be treated differently for search purposes?

Yes. A user may restrict all searches to, for example, the byline field of a newspaper. This functionality is a generalization of restricting searches to a particular field or set of fields.

3.11 Can custom filters be developed for the search engine?

Yes. Custom filters can be developed for the indexing and searching purposes. Typically, these filters are developed to allow the searching of nonstandard/proprietary data formats in their native form. Such filters are developed to a well defined and documented subroutine interface.

These filters are used as part of the indexing process to "skip over" any special markup and formatting codes that are in the documents being indexed. They are also used during the search phase for the same purposes.

3.12 Can the system be setup to ignore search hits to duplicate documents?

Yes. The duplicate articles can be marked and the hits emanating from these documents can be subtracted from the total number of hits.

3.13 Can the system provide hyper-text links and links to photo and sound archives?

Support for multimedia (sound, image, etc.) and hyper-text links is provided for by the Open Text system. Handlers for these data types are invoked by filters which intercept the appropriate references. For example, sample filters are provided in source code form, that intercept image references and invoke the appropriate image viewer and request the image viewer display a particular image. These filters are simple and straight forward to implement and configure. Using

these samples, an audio playback program could similarly be invoked to playback "a sound bite" or an audio/video program to play a "video clip".

4.0 User and Session Limitations

4.1 Access control must be supported to regulate read and write access to various databases.

The Open Text system uses standard operating system access control mechanisms to provide read or read/write privileges to users or groups of users. By using standard OS access control policies administrative policies may be implemented to give, for example:

- users read only access to some databases
- special users read only access to all databases
- librarians read and update access to databases within their area of responsibility
- system administrators read and update access to all databases

4.2 Does the system provide accounting information?

Open Text provides accounting data which includes:

- userid of the user
- amount of CPU time used
- databases accessed
- documents retrieved
- format of the various documents retrieved
- source of the document (if documents have a source field, e.g. engineering, news wire, etc.)

Open Text's modular architecture provides the hooks to insert additional filters which gather specialized accounting information.

5.0 Search Capabilities

5.1 Linear Search

5.1.1 Case Sensitive Searching

The PAT engine provides general support for character mappings, including mapping upper to lower case characters (case insensitive) and no mappings (case sensitive).

5.1.2 Verbatim searching

The PAT engine finds all occurrences of the exact character string typed by the user. Since the PAT engine is based on a string searching algorithm, there is no need for stop words. While we do support stop words, there are absolutely no performance advantages. In fact, we consider "stop words" to be not a feature, but a bug which forces users to exclude valuable contextual data for the sake of performance. For example, PAT will find all occurrences of "to be or not to be" (a phrase in which each word is traditionally a stop word for performance reasons) in the same time as all occurrences of the relatively rare word "renaissance".

5.1.3 Regular Expression Searches

The current release of PAT, Release 4.1, supports trailing wild cards (.* in regular expression notation and * in DOS notation). Thus words like "catastrophe", "catalyst" and "caterpillar" can be found when searching for "cat".

PAT 5.0 will support full regular expressions, allowing internal and left hand wild cards and single character wildcards such as "*t*pill?r" acting as a match to the word "caterpillar". The system will also support queries that are familiar to regular expression users, such as "[a-f]b*p+".

5.2 Boolean Searching

5.2.1 AND, OR and NOT operators

The PAT engine supports a generalization of these operators. Since all results are returned as sets, operators for set intersection, addition and subtraction are provided to allow flexible manipulation of result sets.

5.2.2 Nested Grouping of Operators

These operators may be parenthetically grouped and nested to an infinite level of nesting.

5.3 Proximity Searching

5.3.1 Adjacency

Adjacency operations such as finding occurrences of the words "the", "association" and "of" when they form the phrase "the association of" are done automatically. PAT's string search algorithm allows phrases to be found without requiring special operators, and without any performance degradation.

5.3.2 Bidirectional Adjacency

The phrases "big boat" and "boat big" can be readily found, since Open Text uses a phrase searching algorithm. No special operators are required.

5.3.3 Directional Proximity

The PAT engine provides a "followed by" operator, which allows searches such as "software fby hardware", which finds occurrences of "software" followed by "hardware" within a specified threshold proximity.

5.3.4 Bidirectional Proximity

The PAT engine also provides a "near" operator which can be used in queries such as "software near hardware". This will find all occurrences of "software" which have the word "hardware" either to their left or right (within a specified maximum distance).

5.3.5 Proximity within an element

The Open Text query language supports a high degree of element handling, including searches which find both "hardware" and "software" within the same element (e.g. Paragraph), and searches which find "hardware" and "software" in the same element and within a specified distance of each other.

5.3.6 Statistically significant proximity (signif)

An additional text handling API command is signif, which is a very powerful command used for finding frequently occuring words and phrases in the database and its structures. One illustration of its use might be in answering the question "what words/phrases most frequently follow the phrase 'the association of'?"

5.4 Wildcard support

As described in the section entitled "Regular Expressions", the current release of the PAT engine supports trailing wildcards. PAT 5 will include full regular expression support including internal and left hand multicharacter wildcards and single character wildcards.

5.5 Searching for a word and its synonyms

The PAT engine provides thesaurus support. The user may define a local thesaurus with its own special synonym relations. The format of the PAT thesaurus file format is well documented and in ASCII text, allowing third party thesauri to be adapted for use by the PAT engine.

5.6 Acronym Searches

The PAT engine's string search algorithm provides strong support for acronym searching. Since performance does not degrade when searching commonly occurring words (that in most systems need to be declared as stop words to maintain search speeds), searches for acronyms such as F.A.A. can proceed without problems. This is a problem in most systems, since punctation marks are typically undergo a logical mapping to spaces to allow convenience in searching. This logical mapping typically results in "F.A.A." being viewed (for search purposes) as "F A A", i.e. with spaces between each of the characters. Since the letter "A" by itself occurs very often in the english language, these traditional systems require that "A" be a stop word - thus causing problems in searching for acronyms.

Acronyms such as "ATT", "A.T. & T.", "AT&T" and "A.T. & T." can be treated as equivalent by using PAT's thesaurus feature.

5.7 Accounting for spelling mistakes in word.

Spelling mistakes and searching-with-errors functionality can be added to the current release of

PAT by using customized input/output filters. The next release of PAT will include built-in support for searching-with-errors. This feature will allow the user to specify the number of mismatches that should be allowed in a query. A mismatch can be an insertion of an extra character, a deletion of a required character, a replacement of a character or transpositions of characters in words.

5.8 Weighting Search Terms

The PAT engine supports the weighting of search terms and the ranking of documents according to highest and/or lowest rank received. The ranking support goes beyond traditional methods to also allow the ranking of documents based on ranks assigned to subfields/elements within these documents.

Complex queries may also be expressed and different subcomponents of these queries may be assigned different weights. In this scenario, the documents may be ranked by the overall score of the complex query, which draws its overall score from the weights assigned to query subcomponents.

Since querying documents by date values is simply another query, documents may also be ranked by date queries with higher weighting given to most recent or least recent documents.

5.9 Date Searches

The Open Text system supports date searches within a specified range of dates, before a given date, or after a given date.

5.10 Restricting Searches to the Current Retrieval Set

All PAT queries may be restricted to any set. This flexibility allows the restricting set to be any query set, including the most recent (i.e. current set), thereby permitting multiple iterations on extracted subsets of the database. These subsets can be defined down to any level of granularity of data element.

5.11 Ability to search on whether a field is empty or not.

Queries can be expressed to find documents in which a specific field is present (or not) and also to find documents in which a field is present but empty.

5.12 Ability to return a document count without retrieving a document.

The system, by default returns a match count rather than the documents, thus preventing excessive use of network bandwidth and also allowing user interfaces to operate in a client/server configuration over a modem line.

6.0 Retrieval

The Open Text software supports screen display, printing information and downloading information to another computer. The protocol used for the download is not restricted in any way by the search and retrieval software. Cut and paste between windows is also supported.

Documents can be printed/downloaded either one at a time or in groups, depending on the configuration of the user interface. The user is also given access to summary information in order to decide whether or not to view a document. The summary information for a document may be composed of any fields within that document. By viewing the summary information before selecting a document, the user can receive a high level view of the search results. The summary list approach has the added advantage of conserving network bandwidth and not forcing users to download documents in their entirety before viewing them.

6.1 How can information be saved?

Information to be saved may be either downloaded to the local client or exported to an arbitrary application. In the event that the information is to be saved on another system (e.g., sent on another computer, stored in a relational database, sent to a publishing engine, etc.), it is straightforward to route the data to an application/script which then transfers the data to the appropriate system.

6.2 Can queries be run in batch mode?

The PAT engine can accept queries from a file and run them in batch mode. This approach typically is used for "watchdog" applications in which a predetermined "query profile" must be run against an incoming data stream. In such an application, the successful results can then be routed to the user to whom the profile belongs. This delivery routing can be handled by any appropriate process, including email.

7.0 Text Presentation

7.1 Summary lists must be used to display the text.

Open Text user interfaces allow the display of documents to be based on a summary list. The summary list can either be based on Key Word in Context (KWIC), which shows context surrounding the search term, or a collection of document fields, for example Dateline/Headline in a newspaper database.

7.2 How can documents be viewed?

Documents may be viewed either in their entirety or with specific fields suppressed. The suppression of fields allows for a high level of data abstraction, such as Headlines only in a newspaper application, which permits users to ignore irrelevant details of the data. Users can page up and down in the text being viewed and view selected fields/elements from a particular document. If "page" is defined as a field, then selected pages may also be viewed.

By viewing selected fields of a document, users can rapidly scan through the relevant portions of documents without being forced to export the entire document for viewing.

Open Text provides a "WYSIWYG" text viewer called PowerView (Lector) which is available on X-Windows and MS Windows. The "WYSIWYG" presentation is fully user definable based on style sheets which can be tailored to display the document (or field) in any manner. The user can define many different style sheets for the same class of documents and apply the appropriate style sheet based on their viewing preference.

8.0 Additional Features

8.1 SGML Support

The PAT Server provides high-performance built-in support for SGML structures. In the case where the SGML data is stored in "canonical" or fully-expanded form, the Structure Index Builder can create indices simply and efficiently.

However, when the file is real SGML - validated against a Document Type Definition (DTD) - that DTD contains valuable information about the structures to be found in the text. Furthermore, SGML allows omission or shortening of the tags which mark the structures.

To automate the processing of real SGML, Open Text provides a facility which reads a DTD and processes text against it, automatically writing indices for all the elements and attributes in the text.

This enables simple one-step loading of any SGML instance into a ready-to-use database. It should be noted that Open Text leaves the SGML data in its original form.

8.2 Heterogenous document formats, multi-media hooks & hierarchical searches

Please see the General Requirements section of this document for detailed responses to these issues.

8.3 Protection from specific fields within the database.

This functionality can be customized either through the user interface or through customized I/O filters for the PAT engine.

8.4 International Support

Open Text software provides support for foreign languages such as German, French and other ISO Latin languages. Our Japanese customers also use the PAT engine to search multibyte character sets such as those for the Kanji language. All menu systems and help text files are stored in open file formats and are easily modified as required.

9.0 User Interfaces

Open Text's "out of the box" client software consists of query user interfaces and the PowerView display software. Two query UIs are provided:

- PanelSearch: for novice and casual users. Available for MS-Windows and, as a character mode interface, for ASCII terminals or emulators.
- PowerSearch: for expert/power users. Available for MS-Windows and X-Windows.

These user interfaces provide access to many of the PAT search engine features. They are highly configurable, based on the elements (fields) present in the various databases being searched. Users may also configure these interfaces to use different aliases, hot keys and mnemonics. All such parameters are configurable through control files which are editable by standard ASCII text editors. Where possible, standard MS Windows.ini files and X-Windows resource files are used for such parameters.

Developers and integrators that wish to develop customized user interfaces use the same API which is utilized by PanelSearch and PowerSearch. A set of exhaustively documented sample source code for user interfaces is provided by Open Text to assist user interface developers in getting started. Developers of Microsoft Windows user interfaces may also obtain source code for VisualBasic objects that form the basis of PanelSearch.

Open Text software also allows for the definition and invocation of macros for commonly used queries.

9.1 Command line and query editing

The Open Text user interfaces allow query editing in the normal Windows or Motif style. (When command line interfaces are used, the user has full flexibility to edit the command line using standard terminal cursor keys.

9.2 System and user defaults

System parameters are all specified in control files. These control files may be either used on a system wide or a per user basis, thus allowing a range of applicability. Users may also define their own "per session" settings.

9.3 Customizable display characteristics

Characteristics such as display style sheets, which determine the displayable fields, and summary list criteria are all stored in configuration files. These files persist across sessions and may be either user or installation specific.

9.4 Saving retrieved documents

Users may save lists of retrieved documents across sessions. These documents may be either stored by content (a copy of the document is saved) or by reference (a pointer to the document is saved).

9.5 Query interrupts

The user can interrupt queries at any time, although this is seldom necessary due to the speed of

the PAT search engine.

9.6 Selection Limits

Users may specify retrieval limits, to restrict the number of articles which can be returned.

9.7 Status information

The Open Text system provides clear status feedback about why queries were satisfied and to what intermediate results they refer.

9.8 User defined search space

The user may specify their own search space in terms of a variety of criteria. These include databases, fields, contents of fields (e.g. dates) and other full text criteria.

9.9 Database Maintenance

Open Text provides a number of tools to be used with the PAT search engine for database maintenance purposes. These tools allow data to be added to the database and delete from the database. A database configuration can be set up so that the system is completely searchable while it is being updated. That is, users will not be locked out when data is being added and system maintenance is in progress.

9.10 Capacity Planning

Open Text provides a load simulator tool which can be used to simulate large user populations issuing various levels of query traffic to examine issues such as network bandwidth, server loads, disk contention, etc., and thereby assist in the capacity planning exercise.

"Intentional Blank Page"

"Intentional Blank Page"